APPLIED MATHEMATICS
UNIVERSITY of WASHINGTON

High Performance Scientific Computing

AMath 483/583 Class Notes

Spring Quarter, 2011

Useful gfortran flags

gfortran has many different command line options (also known as *flags*) that control what the compiler does and how it does it. To use these flags, simply include them on the command line when you run gfortran, e.g.:

```
$ gfortran -Wall -Wextra -c mysubroutine.f90 -o mysubroutine.o
```

If you find you use certain flags often, you can add them to an alias in your `.bashrc` file, such as:

```
alias gf="gfortran -Wall -Wextra -Wconversion -pedantic"
```

Output flags

These flags control what kind of output gfortran generates, and where that output goes.

- `-c`: Compile to an object file, rather than producing a standalone program. This flag is useful if your program source code is split into multiple files. The object files produced by this command can later be linked together into a complete program.
- `-o FILENAME`: Specifies the name of the output file. Without this flag, the default output file is `a.out` if compiling a complete program, or `SOURCEFILE.o` if compiling to an object file, where `SOURCEFILE.f90` is the name of the Fortran source file being compiled.

Warning flags

Warning flags tell gfortran to warn you about legal but potentially questionable sections of code. These sections of code may be correct, but warnings will often identify bugs before you even run your program.

- `-Wall`: Short for "warn about all," this flag tells gfortran to generate warnings about many common sources of bugs, such as having a subroutine or function with the same name as a built-in one, or passing the same variable as an `intent(in)` and an `intent(out)` argument of the same subroutine. In spite of its name, this does not turn all possible `-W` options on.
- `-Wextra`: In conjunction with `-Wall`, gives warnings

Table Of Contents

Previous topic

Fortran

Next topic

Fortran subroutines and functions

This Page

Show Source

Quick search

Go

Enter search terms or a module, class or function name.

about even more potential problems. In particular, `-Wextra` warns about subroutine arguments that are never used, which is almost always a bug.

- `-Wconversion`: Warns about implicit conversion. For example, if you want a double precision variable `sqrt2` to hold an accurate value for the square root of 2, you might write by accident `sqrt2 = sqrt(2.)`. Since `2.` is a single-precision value, the single-precision `sqrt` function will be used, and the value of `sqrt2` will not be as accurate as it could be. `-Wconversion` will generate a warning here, because the single-precision result of `sqrt` is implicitly converted into a double-precision value.
- `-pedantic`: Generate warnings about language features that are supported by gfortran but are not part of the official Fortran 95 standard. Useful if you want be sure your code will work with any Fortran 95 compiler.

Fortran dialect flags

Fortran dialect flags control how gfortran interprets your program, and whether various language extensions such as OpenMP are enabled.

- `-fopenmp`: Tells gfortran to compile using OpenMP. Without this flag, OpenMP directives in your code will be ignored.
- `-std=f95`: Enforces strict compliance with the Fortran 95 standard. This is like `-pedantic`, except it generates errors instead of warnings.

Debugging flags

Debugging flags tell the compiler to include information inside the compiled program that is useful in debugging, or alter the behavior of the program to help find bugs.

- `-g`: Generates extra debugging information usable by GDB. `-g3` includes even more debugging information.
- `-fbacktrace`: Specifies that if the program crashes, a backtrace should be produced if possible, showing what functions or subroutines were being called at the time of the error.
- `-fbounds-check`: Add a check that the array index is within the bounds of the array every time an array element is accessed. This substantially slows down a program using it, but is a very useful way to find bugs related to arrays; without this flag, an illegal array access will produce either a subtle error that might not become apparent until much later in the program, or will cause an immediate segmentation fault with very little information about cause of the error.
- `-ffpe-trap=zero,overflow,underflow` tells

Fortran to *trap* the listed floating point errors (fpe). Having `zero` on the list means that if you divide by zero the code will die rather than setting the result to `+INFINITY` and continuing. Similarly, if `overflow` is on the list it will halt if you try to store a number larger than can be stored for the type of real number you are using because the exponent is too large.

Trapping `underflow` will halt if you compute a number that is too small because the exponent is a very large negative number. For 8-byte floating point numbers, this happens if the number is smaller than approximate `1E-324`. If you don't trap underflows, such numbers will just be set to 0, which is generally the correct thing to do. But computing such small numbers may indicate a bug of some sort in the program, so you might want to trap them.

Optimization flags

Optimization options control how the compiler optimizes your code. Optimization usually makes a program faster, but this is not always true.

- `-O` *level*: Use optimizations up to and including the specified level. Higher levels usually produce faster code but take longer to compile. Levels range from `-O0` (no optimization, the default) to `-O3` (the most optimization available).

Further reading

This list is by no means exhaustive. A more complete list of gfortran-specific specific flags is at http://gcc.gnu.org/onlinedocs/gfortran/Invoking-GNU-Fortran.html.

gfortran is part of the GCC family of compilers; more general information on GCC command line options is available at http://gcc.gnu.org/onlinedocs/gcc/Invoking-GCC.html, although some of this information is specific to compiling C programs rather than Fortran.

See also <*http://mpc.uci.edu/man/gfortran.html*>.

Contents | Bibliography | Search | Class Webpage | previous | next | index